Technical Report

IRI-TR-06-09

Probability Density Estimation of the Q Function for Reinforcement Learning IRI Technical Report

Alejandro Agostini
Enric Celaya

Institut de Robòtica i Informàtica Industrial

Abstract

Performing Q-Learning in continuous state-action spaces is a problem still unsolved for many complex applications. The Q function may be rather complex and can not be expected to fit into a predefined parametric model. In addition, the function approximation must be able to cope with the high non-stationarity of the estimated q values, the on-line nature of the learning with a strongly biased sampling to convergence regions, and the large amount of generalization required for a feasible implementation. To cope with these problems local, non-parametric function approximations seem more suitable than global parametric ones. A kind of function approximation that is gaining special interest in the field of machine learning are those based on densities. Estimating densities provides more information than simple function approximations which can be used to deal with the Reinforcement Learning problems. For instance, density estimation permits to know the actual distribution of the q values for any given state-action, and provides information about how many data has been collected in different regions of the domain. In this work we propose a Q-Learning approach for continuous state-action spaces based on joint density estimations. The density distribution is represented with a Gaussian Mixture Model using an on-line version of the Expectation-Maximization algorithm. We propose a method that handles the biased sampling problem with good performance. Experiments performed on a test problem show remarkable improvements over previous published results.

Institut de Robòtica i Informàtica Industrial (IRI)
Consejo Superior de Investigaciones Científicas (CSIC)
Universitat Politècnica de Catalunya (UPC)
Llorens i Artigas 4-6, 08028, Barcelona, Spain
Tel (fax): +34 93 401 5750 (5751)
http://www.iri.upc.edu

Corresponding author:
Alejandro Agostini
tel: +34 93 4015786
agostini@iri.upc.edu
http://www.iri.upc.edu/people/agostini

1 Introduction

Reinforcement learning (RL) in continuous domains requires some form of function approximation to represent the Value function or the Q function, depending on the approach. For non-trivial problems, these functions may be rather complex and can not be expected to fit into a predefined parametric model, especially if the user has not enough a priori knowledge about how such functions should look like. In addition, due to the recursive nature of the value function estimation in reinforcement learning, the function approximation method must be able to cope with non-stationarity, since the value attributed to a state depends on the values assigned to other states, which vary with time as learning takes place. Even more, Reinforcement Learning takes place along trajectories in the state-action space that are dictated by the dynamics of the system and the trade-off between exploitation and exploration. This causes the sampling of the state-action space to be highly biased, a fact that may pose difficulties to some function approximation methods. To deal with these difficulties, local, non-parametric function approximation models like, e.g., radial basis functions, seem more suitable than global parametric approximations.

A kind of function approximation methods that are receiving increasing interest in the field of machine learning are those based on density estimations [2]. Density estimations keep all the information contained in the data, and despite being more demanding than simple function approximation (due to the fact that they embody more information), their use for function approximation has been advocated by different authors [6, 8]. One reason is that simple and well understood tools like the Expectation-Maximization (EM) algorithm [3] can be used to obtain accurate estimations of the density function. Besides this, there are two properties of density estimations that can be useful for RL: First, they allow representing arbitrary multimodal probability distributions of the values taken by the output variable. Second, density estimations provide information about how many data has been collected in different regions of the domain, what can be used to estimate the confidence with which the approximation can be locally credited. This last aspect is important in RL, as this information can be used, for instance, as the basis on which to solve the trade-off between exploitation and exploration.

In the present work, we use a Gaussian Mixture Model [2] to estimate the probability density function in the joint space of state, action, and q value to approximate the q-distribution of a continuous state-action RL problem. To estimate densities in RL we need an on-line version of the EM algorithm. We adapt the on-line EM algorithm for Normalized Gaussian Networks (NGnet) of [11] to the Gaussian Mixture Model case. Afterward, we identify a weakness of this approach, which becomes relevant when sampling is biased, as is the case in RL, and we introduce a forgetting factor that is dependent on the activation of each mixture component, achieving a more local updating of the density estimation. We performed experiments on a test problem and compare our results with those obtained by [10], showing the improvement in performance of our approach.

The rest of the paper is organized as follows: Section 2 introduces the basics of RL and establishes the bases for its formulation with GMM. Section 3 introduces the concepts of GMM for multivariate density estimation, and the EM algorithm in its batch version. In Section 4 we define the on-line EM algorithm for the GMM. In Section 5 we present our RL approach using density estimation of the Q-value function, involving action evaluation and action selection. Section 6 describes an example application to show the feasibility of the approach. We conclude in Section 7 with a discussion of the proposed approach.

2 The Reinforcement Learning Problem

Reinforcement learning is a learning paradigm in which an agent must improve its performance by selecting actions that maximize the accumulation of rewards provided by the environment [13]. At each time step, the agent observes the state s_t and chooses an action a_t according to its policy $u(s)$. The environment changes to state s_{t+1} in response to this action, and produces an instantaneous reward r_t. One of the most popular algorithms used in RL is Q-Learning [14], which uses an action-value function $Q(s,a)$ to evaluate the expectation of the maximum future cumulative reward that will be obtained from various executions of an action a in a situation s. Q-learning uses a sampled version of the Bellman optimality equations [1] to estimate instantaneous q values,

$$q(s,a) = r(s,a) + \gamma \max_{a'} Q(s',a') \tag{1}$$

where $r(s,a)$ is the immediate reward obtained from executing action a in situation s, $\max_{a'} Q(s',a')$ is the maximum estimated reward in the next observed situation s', and γ is a discount factor, with values in [0,1], that regulates the importance of future rewards. At a given stage of the learning, the temporary policy can be derived from the learned q-values as,

$$u(s) = \max_{a} Q(s,a) \tag{2}$$

In an actor/critic architecture, as that used in [10], the policy is also learned and stored in a separate function. Actions are dictated by the policy explicitly represented by the actor, and not computed from the maximization in 2. Despite this computational advantage, the learning of an actor may slow down convergence, since then the learning of the Q-function becomes on-policy instead of off-policy, and both functions, actor and critic, must adapt to each other to reach convergence. In our implementation we avoid the use of an actor, and thus we must face the problem of maximizing the $Q(s,a)$ function in (2).

The basic formulation of Q-learning assumes discrete state-action spaces and the Q-function is stored in a tabular representation. For continuous domains a function approximation is required to represent the Q-function and generalize between similar situations. In next sections we present our proposal for function approximation using densities estimations.

3 The Gaussian Mixture Model

In this work we select a mixture of multivariate Gaussians, known as Gaussian Mixture Model [2], for the density representation,

$$p(\mathbf{x}_t|\theta) = \sum_{i=1}^{K} \alpha_i \mathrm{N}(\mathbf{x}_t|\theta_i) \tag{3}$$

where α_i is the prior probability of Gaussian i, $P(i)$, of generating any sample $\mathbf{x}_t$, usually denoted as the mixing parameter. $\theta_i = \{\mu_\mathbf{i}, \sum_\mathbf{i}\}$ are the parameters of Gaussian i, and $\theta = \{\{\alpha_1, \mu_\mathbf{1}, \sum_\mathbf{1}\}, ..., \{\alpha_\mathbf{K}, \mu_\mathbf{K}, \sum_\mathbf{K}\}\}$ is the whole set of parameters for the mixture. By allowing the adaption of the number K of Gaussians in the mixture, a rich class of density distributions can be approximated. The parameters of the model can be estimated using a maximum-likelihood estimator (MLE). Given a set of samples $\mathbf{X}$, the likelihood function is given by,

$$L[X;\theta] = \prod_{t=1}^{N} p(\mathbf{x}_t|\theta) \tag{4}$$

The maximum-likelihood estimation of the model parameters is the one that maximizes the likelihood (4) for the data set $\mathbf{X}$. Direct computation of the MLE requires complete information about which mixture component generated which instance. Since this information is missing the EM algorithm, described in the next section, is usually employed.

3.1 The Expectation-Maximization algorithm

The Expectation-Maximization algorithm (EM) [3] permits to estimate the parameters that maximize the likelihood function 4 with some missing variables. The EM method first produces an estimation of the expected values of the missing variables using initial values of the parameters to be estimated, and then computes the MLE of the parameters given the expected values of the missing variables. The former procedure is denoted as the E step while the later is the the M step. This process is repeated iteratively until a convergence criterion is fulfilled. The convergence criterion could be either related to a threshold for the variation of the likelihood function 4 or a threshold for the variation of the estimated parameters θ [7].

In this section we briefly describe how EM is applied for the case of GMM parameters estimation. The process starts with an initialization of the mean vectors and covariances matrices of the Gaussians. The E step consists in the calculation of the probability $P(i|\mathbf{x}_t)$ for each component i of generating instance $\mathbf{x}_t$ that we denote by $w_{t,i}$,

$$w_{t,i} = P(i|\mathbf{x}_t) = \frac{P(i)p(\mathbf{x}_t|i)}{\sum_{j=1}^{K} P(j)p(\mathbf{x}_t|j)} = \frac{\alpha_i N(\mathbf{x}_t|\mu_i, \sum_i)}{\sum_{j=1}^{K} \alpha_j N(\mathbf{x}_t|\mu_j, \sum_j)} \tag{5}$$

where $t = 1, .., N$ and $i = 1, .., K$. The maximization step consists in computing the MLE using the estimated $w_{t,i}$. It can be shown [5] that, for the case of a GMM, the mixing parameters, the means, and the covariances are given by,

$$\alpha_i = \frac{1}{N} \sum_{t=1}^{N} w_{t,i} \tag{6}$$

$$\mu_i = \frac{\sum_{t=1}^{N} w_{t,i}\mathbf{x}_t}{\sum_{t=1}^{N} w_{t,i}} \tag{7}$$

$$\Sigma_i = \frac{\sum_{t=1}^{N} w_{t,i}(\mathbf{x}_t - \mu_i)(\mathbf{x}_t - \mu_i)'}{\sum_{t=1}^{N} w_{t,i}} \tag{8}$$

4 On-line EM

Estimating a probability density distribution by means of the EM algorithm involves the iteration of E and M steps on the complete set of available data, that is, the mode of operation of EM is in batch. However, in RL, sample data are not all available at once: they arrive sequentially, following some exploration/exploitation strategy based on the current model estimation. This prevents the use of the raw EM algorithm, and requires an on-line version of it. Some on-line, or sequential, EM algorithms have been proposed for the Gaussians Mixture Model [9, 12].

In [11] an on-line EM algorithm for the NGnet is presented. Here we adopt a similar approach, but adapted to the case of GMM. It consists in performing an E step and a M step after

the observation of each individual sample. The E step does not differ from the batch version (equation 5), though it is only computed for the new sample. For the M step, the parameters of all mixture components are updated with the new sample. For this, we first define the following time-discounted weighted sums,

$$W_{t,i} = [[1]]_{t,i}\,, \tag{9}$$

$$X_{t,i} = [[\mathbf{x}]]_{t,i}\,, \tag{10}$$

$$(XX)_{t,i} = \left[\left[(\mathbf{x}-\mu_i)(\mathbf{x}-\mu_i)'\right]\right]_{t,i} \tag{11}$$

where,

$$[[f]]_{T,i} = \sum_{t=1}^{T}\left(\prod_{s=t+1}^{T}\lambda_s\right)(f_t w_{t,i}) \tag{12}$$

where λ_t, which ranges in [0,1], is a time dependent discount factor introduced for forgetting the effect of old, less accurate values. [1]

The interpretation of $W_{t,i}$ is the number of samples assigned to unit i with discount. $X_{t,i}$ is the discounted accumulation of the samples provided to Gaussian i which is used to derive its mean vector μ_i. $(XX)_{t,i}$ is the accumulation of the squared distances of the observed samples with respect to μ_i, used to calculate the covariance matrix Σ_i.

When a new sample $\mathbf{x}_t$ arrives, these accumulators are updated with the step-wise incremental formula,

$$[[f]]_{t,i} = \lambda_t\,[[f]]_{t-1,i} + f_{t,i}w_{t,i} \tag{13}$$

Then, the GMM estimators can be obtained as,

$$\alpha_{t,i} = \frac{W_{t,i}}{\sum_{j=1}^{K} W_{t,i}} \tag{14}$$

$$\mu_{t,i} = \frac{X_{t,i}}{W_{t,i}} \tag{15}$$

$$\Sigma_{t,i} = \frac{(XX)_{t,i}}{W_{t,i}} \tag{16}$$

4.1 Dealing with Biased Sampling

In the incremental formula (13) the factor λ_t is used to progressively replace (forget) old data by new arrived ones in a smooth way. This is the desired effect when data are presented in a statistically unbiased way, so that all past entries are equally forgotten at the arrival of each new datum. However, in RL data are presented along the system trajectory, and is particularly biased toward the good-valued regions due to the exploitation requirement. That is, convergence regions are highly sampled compared to others, increasing their densities, but this is at the expense of forgetting data and lowering density in the entire domain. This is undesirable since, statistics in regions with low q values, and hence sparsely sampled, will get their data forgotten. To avoid this we modified the updating formula (13) in this way,

$$[[f]]_{t,i} = \lambda_t^{w_{t,i}}\,[[f]]_{t-1,i} + f_{t,i}w_{t,i} \tag{17}$$

[1] In [11] time-discounted weighted means, instead of sums, are defined by normalizing the sums (12) with a factor $\eta_T = \left(\sum_{t=1}^{T}\prod_{s=t+1}^{T}\lambda_s\right)^{-1}$. However, in the expressions of the estimators, all these η_T cancel out, and we can skip its calculation.

With this updating formula the amount of updating performed is not only variable in time but also in space, according to the local nature of the Gaussians approximation. The power $w_{t,i}$ prevents an unbalanced updating of the parameters of the Gaussians which are not responsible of generating the observed values.

5 Q-Learning using Density Estimations

In this section we present the Q-Learning approach using density estimation with a GMM. For the formulation of RL an instance $\mathbf{x}_t$ corresponds to a sample in the RL joint space, $\mathbf{x}_t = (\mathbf{s}, a, q)$.

5.1 Action Selection

Action selection is made according to the policy defined by eq (2), where the $Q(s, a)$ is obtained from the current density estimation as we describe next. Given the GMM,

$$p(\mathbf{s}, a, q) = \sum_{i=1}^{K} \alpha_i \mathrm{N}(\mathbf{s}, a, q | \mu_i, \Sigma_i) \tag{18}$$

where the covariances Σ_i and means μ_i can be decomposed as,

$$\mu_i = \begin{pmatrix} \mu_i^{(\mathbf{s},a)} \\ \mu_i^{q} \end{pmatrix} \tag{19}$$

$$\Sigma_i = \begin{pmatrix} \Sigma_i^{(\mathbf{s},a)(\mathbf{s},a)} \Sigma_i^{(\mathbf{s},a),q} \\ \Sigma_i^{q,(\mathbf{s},a)} \Sigma_i^{qq} \end{pmatrix} \tag{20}$$

The probability distribution for q, $p(q|\mathbf{s}, a)$ for the given state s and a tentative action a, can be obtained in the following way,

$$p(q|\mathbf{s}, a) = \sum_{i=1}^{K} \beta_i(\mathbf{s}, a) \mathrm{N}\left(q | \mu_i(q|\mathbf{s}, a), \sigma_i(q)\right) \tag{21}$$

where,

$$\mu_i(q|\mathbf{s}, a) = \mu_i^q + \Sigma_i^{q,(\mathbf{s},a)} \left(\Sigma_i^{(\mathbf{s},a)(\mathbf{s},a)}\right)^{-1} \left((\mathbf{s}, a) - \mu_i^{(\mathbf{s},a)}\right) \tag{22}$$

$$\sigma_i^2(q) = \Sigma_i^{qq} - \Sigma_i^{q,(\mathbf{s},a)} \left(\Sigma_i^{(\mathbf{s},a)(\mathbf{s},a)}\right)^{-1} \Sigma_i^{(\mathbf{s},a),q} \tag{23}$$

$$\beta_i(\mathbf{s}, a) = \frac{\alpha_i p(\mathbf{s}, a | \mu_i^{(\mathbf{s},a)}, \Sigma_i^{(\mathbf{s},a)(\mathbf{s},a)})}{\sum_{j=1}^{K} \alpha_j p(\mathbf{s}, a | \mu_j^{(\mathbf{s},a)}, \Sigma_j^{(\mathbf{s},a)(\mathbf{s},a)})} \tag{24}$$

from (21) we can obtain the conditional mean and covariance, $\mu(q|\mathbf{s}, a)$ and $\sigma^2(q|\mathbf{s}, a)$ respectively,

$$\mu(q|\mathbf{s}, a) = \sum_{i=1}^{K} \beta_i(\mathbf{s}, a) \mu_i(q|\mathbf{s}, a) \tag{25}$$

$$\sigma^2(q|\mathbf{s}, a) = \sum_{i=1}^{K} \beta_i(\mathbf{s}, a)(\sigma_i^2(q) + (\mu_i(q|\mathbf{s}, a) - \mu(q|\mathbf{s}, a))^2) \tag{26}$$

To select an action we use the policy function (2) where the maximum is taken form a finite set of action values, a_n , regularly sampled along its range. The $Q(s, a_n)$ is obtained stochastically from a normal distribution with mean (25) and variance (26). This provides an adaptive form of exploration that increases the probability of executing exploratory actions when predictions are less certain.

5.2 Statistic Updating

To update the density model with the new experience we must provide the sample $\mathbf{x}_t = (\mathbf{s}_t, a_t, q_{t+1}(s,a))$, where the $q_{t+1}(s,a)$ is given by (1). This equation involves again the estimation of the maximum value $\max_a Q(s,a)$. We proceed in the similar way as in action selection by computing Q for a finite number of a_n, but in this case, we just take as $Q(s,a)$ the expected value given by (25). The same procedure is used to define the policy when exploitation without exploration is desired.

5.3 Active Unit Generation

The approximation capabilities of a GMM depend on the number K of Gaussians of the mixture. Since we can not determine the most appropriate number beforehand, the number of Gaussians need to be regulated on-line.

The conditions under which a new Gaussian is generated are the following. First, when the estimation error (of the q values) is larger than value δ, and second, when the density of samples in the experienced instance is below a threshold ρ. This implies that a Gaussian is generated in regions devoided of samples only when the generalization performed is insufficient. Criteria for Gaussian generation are then expressed as,

$$p(\mathbf{s}, a, q) = \sum_{i=1}^{K} \alpha_i \mathrm{N}(\mathbf{s}, a, q | \mu_i, \Sigma_i) \leq \rho \tag{27}$$

for the density evaluation, and,

$$(q(s,a) - \mu(q|\mathbf{s}, a))^2 \geq \delta \tag{28}$$

for the maximum allowed estimation error.

Whenever both criteria are fulfilled a Gaussian is generated with parameters given by,

$$W_{K+1} = 1 \tag{29}$$

$$\mu_{K+1}(\mathbf{s}, a, q) = (s_t, a_t, q(s,a)_{t+1}) \tag{30}$$

$$\Sigma_{K+1} = C \; diag\{d_1, ..., d_D, d_a, d_q\} \tag{31}$$

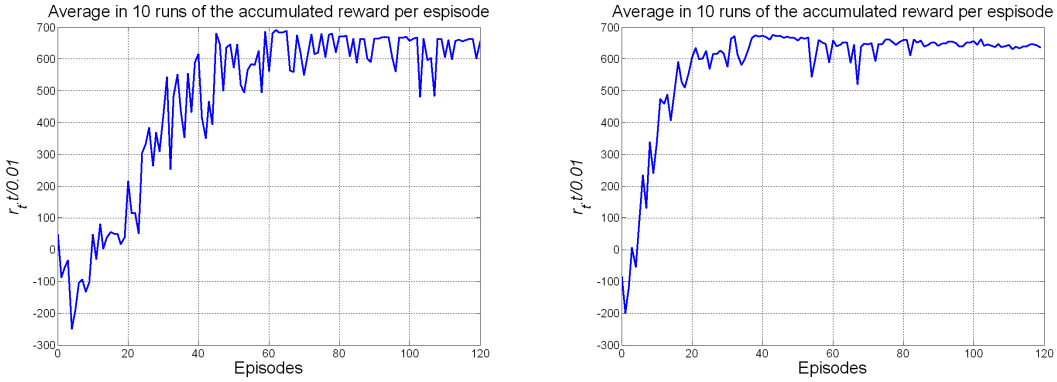
where d_i is the total range of the variable i and is D the dimensionality of the state space. C is a positive value to regulate the dispersion of the new Gaussian.

It is convenient to initialize the covariances to cover a local region of the sampled point in consonance of the local nature of the approximation.

6 Experiments

To evaluate the method we select a benchmark application of an inverted pendulum with limited torque [4] also employed in the work of [10]. The task consists in swinging the pendulum until the upright position is reached and held. The optimal policy for this control is not trivial as the limited torque forces the controller to swing the pendulum several times until the kinetic energy of the pendulum is larger than the load torque so the upright position can be reached.

The state space is two-dimensional and is configured by the angular position θ and angular velocity $\dot{\theta}$ variables, $\mathbf{s} = (\theta, \dot{\theta})$, where θ ranges in the interval $[-\pi, \pi]$. Thus, the density is estimated in a four-dimensional joint space $\mathbf{x}$=$(\theta, \dot{\theta}, a, q)$. As the reward signal we simply take the height of the tip of the pendulum $h = cos(\theta)$ which ranges in the interval $[-1, 1]$, and the discount coefficient γ in equation 1 is set to 0.99.



(a) with λ_t (b) with $\lambda_t^{w_{t,i}}$

re 1: Average of the accumulated reward per episode performed over 10 experiments. (a) Updating using λ_t. (b) Updating u

The setting of our system consists in the following. We provide the system with 20 initial Gaussians. The elements of the mean μ_i of the mixture component i are selected randomly, except for the q variable that is initialized to the maximum possible value to favor exploration of unvisited regions. The initial covariance matrices Σ_i are diagonal and the variance for each variable is set to the range of the variable. This is actually a wide covariance that provides a coarse estimation of q at the initial stages of learning. Finally, the initial number of samples for each Gaussian i, W_i, is set to 0.1. This small value makes the component i to have little influence in the inference while there is no, or little, updating.

The discount coefficient λ for the cumulative values calculations (section 4) takes values from the equation,

$$\lambda_t = 1 - 1/(at + b) \tag{32}$$

where b regulates the initial value of λ (when t=0) and a determines its growth rate toward 1. In our experiments we set a to 0.001 and b to 10 so as to permit a flexible adaptation to the non-stationarities mainly at the beginning of the learning process.

We compare the performance of our approach with the method proposed in [10] as it is the most related to our approach. As in [10] we performed the experiments using episodes of 7 seconds with a simulation interval of 0.01 seconds. After each episode the pendulum is randomly placed inside an arch centered in the upright position. The length of the arch is steadily incremented with each episode.

Figure 1 shows a comparison between the on-line EM using the conventional discount factor λ_t, and the new updating proposal $\lambda_t^{w_{t,i}}$. The variable evaluated is the average of the total accumulated reward per episode for 10 experiments of 120 episodes each, exploiting the policy learned so far. Results show that the new proposal achieves convergence much faster, and with a more stable steady state of the learning process. The spikes present in figure 1 (a) are caused by the discount factor λ_t which is equally applied in all regions of the domain. This produces a "forgetting" of the experiences made so far in regions far from the sampled situation without any replacement with new information. This effect is greatly reduced when the factor $\lambda_t^{w_{t,i}}$ is used, as shown in figure 1(b).

To compare with the work of [10] we compute the average number episodes and the number Gaussians needed to attain convergence. In the case of updating with λ, convergence occurs at about 44 episodes with a mean number of Gaussians of 27. In the second case, with an updating using $\lambda_t^{w_{t,i}}$, the convergence takes place in an early stage with only 20 episodes, with a number of Gaussians of 35. In [10], the reported number of episodes to reach convergence is

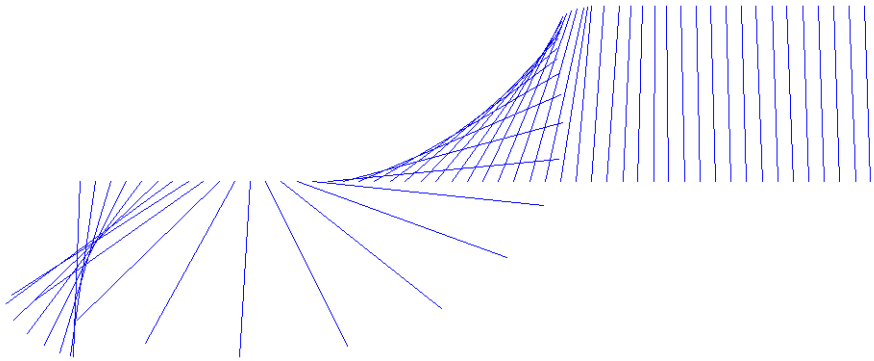


Figure 2: A stroboscopic sequence obtained from placing the pendulum in the downright position.

40, and the number of components of the NGnet needed to approximate the critic is 108 (for a fair comparison, components of the actor are not considered). For informative sake, our best results in the 10 runs using updating with $\lambda_t^{w_{t,i}}$ converged in 4 espisodes with 21 Gaussians.

To illustrate the performance of the control reached we present in figure 2 a control from the difficult position of the pendulum hanging down. In general, for all the experiments, the pendulum could reach and held the upright position from almost any initial place from the very first stage of the convergence.

7 Conclusions

We proposed a new approach for Q-Learning in continuous state-action spaces using a multivariate density model, which exploits some of the properties of density estimations to deal with the problems inherent to RL. In particular, we have shown how a Gaussian Mixture Model can be used to estimate a probability density function of the experiences obtained in a continuous reinforcement learning problem, and how it can be used for the approximation of the Q function. For the estimation of the Gaussian Mixture Model we adapted the on-line version of the EM originally developed by [11] for the NGnet, in order to apply it to the GMM, and integrated it into the learning algorithm.

Experiments performed with the updating using the non-activation-dependent forgetting factor, λ_t, on a test problem show that our results are well comparable to those obtained by [10]. Furthermore, we noticed the perturbing effects that biased sampling, intrinsic to reinforcement learning, produces in the convergence of the approximation of the Q function, and we proposed an activation-dependent-forgetting factor, $\lambda_t^{w_{t,i}}$, that mitigates this effect. With this modified factor, we obtained results that significantly improve the previous ones.

Since now, we have just tested our approach on deterministic environments, where the use of a density estimation is not fully exploited. We expect to obtain further benefits from this approach, when applying it to non deterministic domains, where the Q distribution may be multimodal, so that it can not be represented by simple function approximation. We would continue exploring how to take more advantages of the rich information contained in density estimations for the problem of RL, like investigating new strategies for exploration-exploitation based on density of samples in the domain.

References

[1] R. Bellman and S. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, New Jersy, 1962.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] A.P. Dempster, N.M. Laird, D.B. Rubin, et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[4] K. Doya. Reinforcement learning in continuous time and space. *Neural Comput.*, 12(1):219–245, 2000.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley and Sons, Inc, New-York, USA, 2001.

[6] M. Figueiredo. On gaussian radial basis function approximations: Interpretation, extensions, and learning strategies. *Pattern Recognition, International Conference on*, 2:2618, 2000.

[7] M. Figueiredo and A. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[8] Z. Ghahramani and M. Jordan. Supervised learning from incomplete data via an em approach. In *Proceeding of Advances in Neural Information Processing Systems (NIPS'94)*, pages 120–127. San Mateo, CA: Morgan Kaufmann, 1994.

[9] R. Neal and G. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*, pages 355–368, Norwell, MA, USA, 1998. Kluwer Academic Publishers.

[10] Masa-aki Sato and Shin Ishii. Reinforcement learning based on on-line em algorithm. In *Proceedings of the 1998 conference on Advances in neural information processing systems (NIPS'99)*, pages 1052–1058, Cambridge, MA, USA, 1999. MIT Press.

[11] Masa-Aki Sato and Shin Ishii. On-line em algorithm for the normalized gaussian network. *Neural Comput.*, 12(2):407–432, 2000.

[12] M. Song and H. Wang. Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. In *Proceedings of SPIE: Intelligent Computing: Theory and Applications III*, pages 174–183, Orlando, FL, USA, 2005.

[13] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[14] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

IRI reports

This report is in the series of IRI technical reports.
All IRI technical reports are available for download at the IRI website
`http://www.iri.upc.edu`.